ATTORNEY DOCKET No.
SUNMP317/P9293

U.S. PATENT APPLICATION

FOR

METHODS FOR EXECUTION CONTROL ACQUISITION OF A PROGRAM AND FOR EXECUTING AN OPTIMIZED VERSION OF A PROGRAM

INVENTOR: Jan Civlin
151 South Bernardo Avenue, Apt. 7
Sunnyvale, California 94086
Citizen of Israel

ASSIGNEE: SUN MICROSYSTEMS, INC.
4150 NETWORK CIRCLE
SANTA CLARA, CALIFORNIA 95054

MARTINE & PENILLA, L.L.P.
710 Lakeway Drive, Suite 170
Sunnyvale, California 94085
Telephone (408) 749-6900

Methods for Execution Control Acquisition of a Program and for Executing an Optimized Version of a Program

by Inventor

Jan Civlin

Cross Reference to Related Applications

This application is related to the following applications: 1) U.S. Application No. 10/371,383, filed on February 21, 2003; and 2) U.S. Application No. ________ (Attorney Docket SUNMP318), filed on August 1, 2003, and entitled "Method for Dynamic Recompilation of a Program." The disclosures of these related applications are incorporated herein by reference.

Background of the Invention

The present invention relates generally to computer systems and, more particularly, to a method for execution control acquisition of a program and a method for executing an optimized version of a program.

In the dynamic recompilation of programs, execution control acquisition from a program that cannot tolerate changes to its code is problematic. A simple solution to the problem would be to use the hardware interrupts to obtain execution control. If, however, the interrupt is set on the timer or other any other system resource that is not relevant to performance metrics, then significant performance degradation may occur due to the many unnecessary interrupts that are bound to the timer and therefore cannot be avoided. As the objective of dynamic recompilation often is to optimize the program for faster execution, such performance degradation is obviously not acceptable.

In view of the foregoing, there is a need for a method of execution control acquisition from a program that does not cause significant performance degradation.

SUMMARY OF THE INVENTION

Broadly speaking, the present invention fills this need by providing, among other things, a method for execution control acquisition of a program that uses a hardware performance counter. Computer readable media containing program instructions for execution control acquisition of a program also are provided.

In accordance with one aspect of the present invention, a method for execution control acquisition of a program is provided. In this method, a program is executed. During the execution of the program, it is determined when a hardware performance counter has reached a threshold. When the threshold is reached, execution control is switched to a dynamic optimizer. Thereafter, an optimized version of the program is executed.

In one embodiment, the execution of the program includes execution of the original binary code for the program. In one embodiment, the execution of the optimized version of the program includes the execution of an optimized binary code file for the program. In one embodiment, the hardware performance counter is one of an instruction counter, a cycle counter, and a cycles per instruction counter. In one embodiment, the hardware performance counter is integrated into a central processing unit.

In accordance with another aspect of the present invention, a method for executing an optimized version of a program is provided. In this method, an optimized version of a program is executed. During execution of the optimized version of the program, an interrupt is received and execution control is returned to an operating system. An original version of the program is then executed. During the execution of the original version of the program, a hardware performance counter is monitored. When the hardware performance counter reaches a threshold during the execution of the original version of

the program, execution control is switched to a dynamic optimizer. Thereafter, the execution of the optimized version of the program is continued as directed by the dynamic optimizer.

In one embodiment, the execution of the optimized version of the program includes the execution of an optimized binary code file for the program. In one embodiment, the execution of the program includes execution of the original binary code for the program. In one embodiment, the hardware performance counter is one of an instruction counter, a cycle counter, and a cycles per instruction counter. In one embodiment, the threshold is set to a value that is indicative of unacceptable performance. In one embodiment, the hardware performance counter is integrated into a central processing unit.

In accordance with yet another aspect of the present invention, computer readable media containing program instructions for execution control acquisition of a program are provided. In one embodiment, the computer readable media includes program instructions for executing a program, program instructions for determining when a hardware performance counter reaches a threshold during the executing of the program, program instructions for switching execution control to a dynamic optimizer when the threshold is reached, and program instructions for executing an optimized version of the program.

In accordance with a further aspect of the present invention, computer readable media containing program instructions for executing an optimized version of a program are provided. In one embodiment, the computer readable media include program instructions for executing an optimized version of a program, program instructions for receiving an interrupt during execution of the optimized version of the program and returning execution control to an operating system, program instructions for executing an

original version of the program, program instructions for monitoring a hardware performance counter during the executing of the original version of the program, program instructions for switching execution control to a dynamic optimizer when the hardware performance counter reaches a threshold during the executing of the original version of the program, and program instructions for continuing the executing of the optimized version of the program as directed by the dynamic optimizer.

It is to be understood that the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention, as claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute part of this specification, illustrate exemplary embodiments of the invention and together with the description serve to explain the principles of the invention.

Figure 1 is a schematic diagram that illustrates an exemplary execution of an optimized version of a program in accordance with one embodiment of the invention.

Figure 2 is a simplified schematic diagram that shows a central processing unit (CPU) provided with hardware performance counters that may be used to implement the methods described herein.

Figure 3 is a flow chart diagram illustrating the method operations performed in the execution of an optimized version of a program in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

Several exemplary embodiments of the invention will now be described in detail with reference to the accompanying drawings.

Figure 1 is a schematic diagram 100 that illustrates an exemplary execution of an optimized version of a program in accordance with one embodiment of the invention. Referring to Figure 1, dynamic optimizer (DO) 102 carries out the execution using original binary code 104 for a program and optimized binary code 106 for the program, as will be described in more detail below. The creation of optimized binary code 106 by an exemplary dynamic optimizer and additional details regarding the functionality of the dynamic optimizer are set forth in commonly owned U.S. Application No. ________ (Attorney Docket SUNMP318), filed on August 1, 2003, and entitled "Method for Dynamic Recompilation of a Program," the disclosure of which is incorporated herein by reference.

As shown in Figure 1, DO 102 has passed execution control to optimized binary code 106 and the optimized binary code is executed along the process flow indicated by arrow A. During execution of optimized binary code 106, an exception occurs and this exception triggers an interrupt that passes execution control to operating system 108 (as indicated by arrow B). In some instances, operating system 108 may not be aware that it took control from optimized binary code 106. Thus, when operating system 108 redirects execution of the program, the operating system may return execution control to original binary code 104 (as indicated by arrow C). As original binary code 104 is executed along the process flow indicated by arrow D, a hardware performance counter will trigger a hardware interrupt, as will be explained in more detail below with reference to Figures 2 and 3. When this interrupt occurs, execution control switches to DO 102 (as indicated by

arrow E) and DO 102 returns execution control to optimized binary code 106 (as indicated by arrow F) for continued execution of the program. Thereafter, execution of optimized binary code 106 continues along the process flow indicated by arrow G. When another exception that passes execution control to operating system 108 occurs, the above-described cycle of events will repeat itself.

As is known to those skilled in the art, hardware performance counters are special purpose registers that are provided in many known computer hardware architectures to enable hardware performance to be monitored. Hardware performance counters typically include two registers: a counter and a threshold. The value of the counter is continually updated by the operating system. The threshold has a constant value that cannot be changed by the operating system, but can be changed manually through the use of appropriate software.

To interrupt a program using a hardware performance counter, the threshold is adjusted to a desired value relative to the current value of the counter. When the current value of the counter reaches the threshold, the hardware performance counter gives rise to an interrupt, which interrupts the execution of the program and passes control to other software, e.g., a dynamic optimizer. Additional details regarding the use of hardware performance counters to interrupt a program are set forth in commonly owned U.S. Application No. 10/371,383, filed on February 21, 2003, the disclosure of which is incorporated herein by reference.

Figure 2 is a simplified schematic diagram 150 that shows an exemplary central processing unit (CPU) provided with hardware performance counters that may be used to implement the methods described herein. As shown in Figure 2, central processing unit (CPU) 152 includes arithmetic logic unit (ALU) 154, floating point (FP) 156, general

purpose registers (GPR) 158, memory 160, clock generator 162, and hardware performance counters 110. It is to be understood that any suitable CPU hardware configuration can be used and, therefore, the CPU may include more or less circuitry than shown in Figure 2. As shown in Figure 2, hardware performance counters 110 are indicated generally by the single block designated by reference numeral 110; however, those skilled in the art will understand that the specific hardware performance counters in an actual CPU are spread across the CPU so that each counter can collect different information. Hardware performance counters 110-1 and 110-2 shown in Figure 2 illustrate locations of exemplary hardware performance counters. Hardware performance counter 110-1, which is used to monitor one aspect of the performance of memory 160, includes load counter 110a-1 and threshold 110b-1. Hardware performance counter 110-2 is coupled to clock generator 162 and includes cycle counter 110a-2 and threshold 110b-2.

Those skilled in the art will appreciate that the specific hardware performance counter used may be varied to suit the needs of particular situations. By way of example, the hardware performance counter may be an instruction counter, a cycle counter, a load counter, a D$-miss counter, an E$-miss counter, or an I$-miss counter. In addition, in hardware architectures that include such a counter, the hardware performance counter may be a cycles per instruction counter. Those skilled in the art will further appreciate that more than one hardware performance counter can be used.

The use of a hardware performance counter to trigger an interrupt will now be explained in terms of a simplified example. In this simplified example the instruction counter is used to trigger the interrupt after 1,000 instructions have been executed. First, a program binds the hardware interrupt with the threshold register and the instruction

counter, sets the value of the threshold to 1,000, and resets the instruction counter. Thereafter, upon execution of the next 1,000 instructions, the threshold becomes equal to the value of 1,000 and the system gives rise to an interrupt. The interrupt routine performs any desired work, e.g., analyzing performance metrics, and then resets the instruction counter. After another 1,000 instructions have been executed, the interrupt process will be repeated again.

Figure 3 is a flow chart diagram 200 illustrating the method operations performed in the execution of an optimized version of a program in accordance with one embodiment of the present invention. The method begins in operation 202 in which an optimized version of a program is executed. In one embodiment, the optimized version of the program is an optimized binary code file for the program. In operation 204, an interrupt is received during execution of the optimized version of the program and execution control is returned to the operating system. The interrupt may be triggered by any exception that may be part of the optimized version of the program or other external code, program, or hardware. The method continues in operation 206 in which an original version of the program is executed. As set forth above in connection with the description of Figure 1, in some instances, the operating system may not be aware that it took control from the optimized version of the program. Thus, when the operating system redirects execution of the program, the operating system may return execution control to the original version of the program. In one embodiment, the original version of the program is the original binary code for the program.

In operation 208, a hardware performance counter is monitored during the execution of the original version of the program. The particular hardware performance counter used may be varied depending upon the performance metric being used to

evaluate the performance of the execution of the original version of the program, and several exemplary hardware performance counters are listed above. If desired, more than one hardware performance counter can be monitored. The use of more than one hardware performance counter may be necessary when the performance metric being used is determined by a calculation based on the values of two or more counters. The hardware performance counter may be monitored by setting the threshold of the counter to a desired value relative to the value of the counter. In one embodiment, the threshold is set to a value that is indicative of unacceptable performance by the original version of the program. When the hardware performance counter reaches the threshold, in operation 210, execution control is switched to a dynamic optimizer. Switching execution control to the dynamic optimizer in this manner avoids unnecessary invocation of the system interrupts when the performance of the original version of the program is satisfactory and thereby helps to maintain the performance of the system on a high level. In operation 212, the execution of the optimized version of the program is continued as directed by the dynamic optimizer. In one embodiment, the dynamic optimizer returns execution control to the optimized binary code file for continued execution of the program. The execution of the optimized version of the program continues until another exception occurs, at which point the above-described sequence of method operations will be repeated.

Those skilled in the art will recognize that the order in which the method operations are performed may be varied from that described herein, e.g., by rearranging the order in which the method operations are performed or by performing some of the method operations in parallel. Further, while the present invention has been described in the general context of an application program that is executed on an operating system in conjunction with a test system, it should be appreciated that the invention may be

implemented with other routines, programs, components, data structures, etc., which perform particular tasks or implement particular abstract data types. In addition, the present invention may be practiced with other computer system configurations including hand-held devices, microprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like.

With the embodiments described herein in mind, it should be understood that the present invention may employ various computer-implemented operations involving data stored in computer systems. These operations are those requiring physical manipulation of physical quantities. These quantities usually, but not necessarily, take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. Further, the manipulations performed are often referred to using terms such as producing, identifying, determining, or comparing.

Any of the operations described herein that form part of the present invention are useful machine operations. The invention also relates to a device or an apparatus for performing these operations. The apparatus may be specially constructed for the required purposes, or it may be a general purpose computer selectively activated or configured by a computer program stored in the computer. In particular, various general purpose machines may be used with computer programs written in accordance with the teachings herein, or it may be more convenient to construct a more specialized apparatus to perform the required operations.

The present invention also can be embodied as computer readable code on a computer readable medium. The computer readable medium may be any data storage device that can store data which can be thereafter be read by a computer system. Examples of the computer readable medium include hard drives, network attached storage

(NAS), read-only memory, random access memory, CD-ROMs, CD-Rs, CD-RWs, magnetic tapes, and other optical and non-optical data storage devices. The computer readable medium also can be distributed over network coupled computer systems so that the computer readable code is stored and executed in a distributed fashion.

In summary, the present invention provides a method for execution control acquisition of a program and a method for executing an optimized version of a program. The invention has been described herein in terms of several exemplary embodiments. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention. The embodiments and preferred features described above should be considered exemplary, with the invention being defined by the appended claims and equivalents thereof.

What is claimed is: